Atrevido: SemiDynamics Out-of-Order RISC-V Core

Roger Espasa, PhD, CEO & Founder
Semidynamics
Semidynamics RISC-V Cores

**AVISPADO 222**
- 2-wide In-Order
- Gazzillion Misses™
- VPU 1.0

**ATREVIDO 222**
- 2-wide Out-of-Order
- Gazzillion Misses™
- VPU 1.0

TODAY’s FOCUS

IP cores available for licensing
ATREVIDO 222

RISCV64GCV

- SV48
- 16KB I$
- Decodes upcoming V1.0 vector spec
- 32KB D$
- Full hardware support for unaligned accesses
- Coherent (CHI)

• Larger BP
• Renaming
• Retirement Logic

Available for licensing
ATREVIDO 222 with OOO VPU (RVV1.0)

RISCV64GCV

- SV48
- 16KB I$
- Decodes upcoming V1.0 vector spec
- 32KB D$
- Full hardware support for unaligned accesses
- Coherent (CHI)

Available for licensing

- Larger BP
- Renaming
- Retirement Logic
SemiDynamics’ VPU

• Implements the RISC-V Vector 1.0 Specification
  • Including lmul, segmented loads
  • No vector atomics currently

• Ready for OOO support
  • Renaming

• Customizable settings
  • VLEN = vreg size = from 128b to 4096b
  • Data path width = from 128b to 512b
  • Fast cross-lane network for slide/rgather/compress/expand

Available for licensing 3 months after V-spec freeze
One slide reminder of RVV

vsetvli  x9 ← x8, e64, m1
vadd.vv  v7 ← v5, v6
VPU Block Diagram

- Lane based organization
- Full cache-line bus from D-cache
- Units per lane
  - FMA
  - INT
  - DIV
  - XL: Cross-lane (rgather, ...)
- Full masking support

Available for licensing 3 months after V-spec freeze
One Two slide reminder of RVV:

\[ \text{vsetvli } x_9 \leftarrow x_8, \text{e}64, \text{m}1 \]

\[ \text{vadd.vv } v_7 \leftarrow v_5, v_6 \]

Assume \( x_8 = 5 \)
Renaming

• To enable OOO execution, you need renaming
• Renaming remaps the virtual program registers to physical registers
• Physical registers are taken from a not-necessarily-sorted FREE LIST
• Example
  • \texttt{vsetvli x9} \leftarrow \texttt{x8, e64, m1} \rightarrow \texttt{vsetvli p2} \leftarrow \texttt{p28, e64, m1}
  • \texttt{vadd.vv v7} \leftarrow \texttt{v5, v6} \rightarrow \texttt{vadd.vv vp11} \leftarrow \texttt{vp2, vp17}
  • \texttt{vadd.vv v7} \leftarrow \texttt{v7, v9} \rightarrow \texttt{vadd.vv vp29} \leftarrow \texttt{vp11, vp28}
• Please note that \texttt{v7} is re-mapped in the second instruction
Renaming and “Background Data”

<table>
<thead>
<tr>
<th>1st VADD writes vp11</th>
<th>2nd VADD writes vp29</th>
</tr>
</thead>
<tbody>
<tr>
<td>H</td>
<td>G</td>
</tr>
</tbody>
</table>

- **IN-ORDER machine**: NO worries, “H”, “G” and “F” are just there
- **OUT-OF-ORDER machine**: “H”, “G”, “F” are in vp11, but not in vp29
  - A COPY IS NEEDED, or
  - VP11 must be an additional source to the second VADD
  - Specially painful if your vector is 4096 bits and your VL only asks for, say, 128b worth of work

\[
vsetvli\ x9\ \leftarrow\ #8,\ e64,\ m1\ \Rightarrow\ vsetvli\ p2\ \leftarrow\ p28,\ e64,\ m1
\]
\[
vadd.vv\ v7\ \leftarrow\ v5,\ v6\ \Rightarrow\ vadd.vv\ vp11\ \leftarrow\ vp2,\ vp17
\]
\[
vsetvli\ x9\ \leftarrow\ #5,\ e64,\ m1\ \Rightarrow\ vsetvli\ p13\ \leftarrow\ p28,\ e64,\ m1
\]
\[
vadd.vv\ v7\ \leftarrow\ v7,\ v9\ \Rightarrow\ vadd.vv\ vp29\ \leftarrow\ vp11,\ vp28
\]

June 28, 2022
Renaming Vector State

• A vector instruction has far more than 3 sources

• \texttt{vadd.vv v4} \leftarrow v8, v12, v0.t
  • V8 in current SEW format
  • V12 in current SEW format
  • V0 in mask format
  • \texttt{LMUL}
  • \texttt{SEW}
  • VL

• In fully OOO machine, need to track them all
  • Simplifications are possible, of course
A short example

• `vsetvli x9 ← x8, e64, m1`
  • `VL = min(x8, MAXVL)`
  • `VTYPE = “e64, m1”`
  • `vsetvli <pVL5, pVTYPE3, p20> ← p12, e64, m1`

• `vadd.vv v7 ← v5, v6`
  • Rename v5 and v6
  • Rename VL
  • Rename VTYPE
  • Potentially 4 sources, optionally one more for v0, and one more for “background”
  • `vadd.vv vp20 ← vp17, vp15, pVL5, pVTYPE3`

• Think of the wakeup logic in an OOO instruction window
  • Must choose between full renaming of state, partial renaming
  • Otherwise, risk stalling the OOO on each vsetvl
Gazzillion Misses™

**Definition** The ability of Semidynamics’ cores to generate a very large number of outstanding memory requests

**Informally** A ton of bandwidth, Good for big data, HPC and AI
Comparison to other cores

Retirement/Issue Width

Outstanding Misses

Atrevido
128+

Avispado
64+

AX25
SCR7
U8
RISC-V Boom
A76
IceLake
POWER9
SMT4

1
2
3
4
5
6
Gazzillion Misses™ good for...

Machine Learning
- MLP
- AE
- DRL
- CNN
- RNN
- LSTM
- GRU

Recommendation Systems
- MLP
- AE
- CNN
- DRL
- RNN
- LSTM
- GRU

Key-Value Stores
- redis
- Memcached
- HPC

Sparse Data / HPC
Gazzillion Misses™ incredibly good for RVV

Can you find a core out there capable of streaming data at over 60 Bytes/cycle?
Semidynamics RISC-V Cores

• Application cores, in-order and out-of-order, coherent
• Great for “Big data”, “high bandwidth” situations
• Great for RVV

• Happy to share PPA with you under NDA
• Happy to customize the core & the vpu for you

Thank you!
Thank you!